MBL 342
Windows Mobile Platform Futures

Samim Erdogan

Program Manager
Mobile Devices Product Group
Microsoft Corporation

Windows Mobile on the Horizon

Improve and complete platform:

- Better tools: Visual Studio 2005
- Net CF V2
- ATL & MFG Runtimes
- Native SQL CE
- All new features & key existing APIs are managed

Extend the development platform:

- Location API
- Notification Broker
- Managed POOM, Telephony, Messaging
- Picture picker, Direct 3D, others

Improve abilities:

- Unified Install; Configuration manager; Pocket Watson
- Security
- Performance

Visual Studio 2005

- New features for Windows Mobile Platform Developers
  - C++ for Devices
  - Managed/Native Projects in one solution
  - COM Interop
  - Debugging
    - Attach debugger to running process
  - Integrated Smartphone support
  - New Designers
  - New Emulator
  - CAB and Setup Projects
  - Remote Tools
    - Registry, process viewer, file viewer, heap

.NET Compact Framework V2

- Managed interfaces for new mobile features
- Most requested desktop features
  - C# Language Features
    - Generics, anonymous methods, iterators
  - UI
    - New Windows Forms controls and properties
    - Support for new devices (screen rotation, hiRes, keyboards
  - Extensibility
    - COM interop, pInvoke marshalling enhancements, hosting the .NETCF CLR

.NET Compact Framework V2

- XML/Data support
  - XML serialization, XML schemas, XPath, SQL CE result set
- New Libraries
  - Serial port, Isolated Storage, Registry, Generic Collections, Cryptography, Sound
- Threading
  - Invoke with parameters, BeginInvoke/EndInvoke, Exception cross-thread GUI calls
- Networking
  - IPv6, Simplified Asyn Web Services model and Web Services SOAP 1.2, Enhanced auth (Kerberos and NTLM), System.Messaging

Runtimes In ROM

- Runtimes in ROM for Future Windows Mobile Platform
  - PocketPC
    - Latest Versions: ATL80, MFC80
    - Previous versions: ATL300, MFC300
  - Smartphone
    - Latest Versions: ATL80, MFC80
  - Supported but not shipped in ROM (SDK)
    - ATL400 (PocketPC and Smartphone)

Native Database Support

- Both SQL Server Mobile and CEDB included
- SQL Server Mobile is engine of choice
  - Rich and Robust Local Database
  - Subset of SQL Server (but not the same)
  - Offline with intermediate connectivity
  - Some porting may be required
- SQL CE Server Components
  - Data storage engine (native OLEDB interface)
  - Query processor
  - Managed provider (ADO.NET)
    - System.Data.SqlServerCE namespace
    - Direct access to data
    - Remote Data Access (RDA)
    - Merge replication

New & Improved APIs

"Help me get work done while in the office and on the go."

Enabling developers to write apps that fulfill this vision:

- **Location API**
- **Notification Broker**
- **Managed POOM, MAPI**
- **Picture picker, Direct 3D, others**

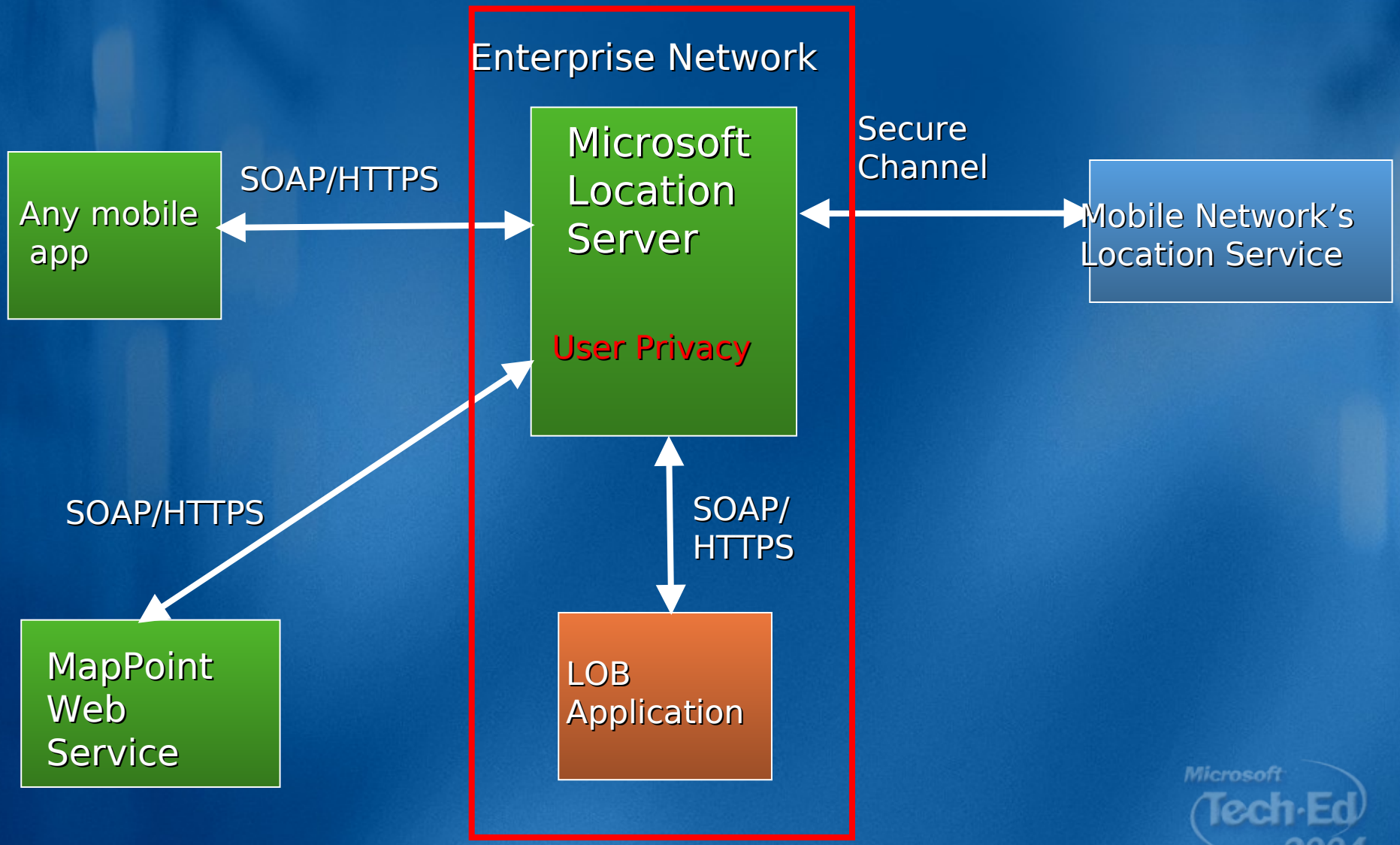
Location API

- Single unified interface for retrieving location information
- App scenarios:
  - Display/use real-time location
  - My location and resources near me
- Native interface
  - Simple Lat/Long only
- Managed interface
  - Richer : lat/long, precision, date/time
  - Enumerates available providers on device
  - Supports local data and web services
  - Integration with MapPoint Location Server

MapPoint Location Server

- Acquires real-time location information from Wireless Operator Networks
  - Where am I? Where are my buddies? Where are my assets?
- Simple SOAP XML API
  - Standards based - Easy for all developers to use, regardless of Language and Operating system
  - Cross Device - Web, Mobile device, desktop applications are all possible with the single API!
- Privacy Framework
  - Gives the Enterprise and end user control

MLS Overview



Notifications Broker

- Unify state and notification architecture across device
  - Query state, register for notification of changes
  - Managed and native interface
  - Persistent (across resets) and transient properties
  - Extensible mechanism for third parties to add state properties
  - Examples: PhoneSignalStrength, Battery, OwnerName, NextAppointmentStartTime, WmplayerCurrentTrack
- Usage Scenarios
  - Launch your application on arrival of a new email or availability of a broadband network connection

Notification Broker

- Change Notification
  - Persistent: App Launching, .NET event
  - Transient: Message queue, Windows message, .NET events
  - Conditional
    - Notify only if condition met
    - DWORD: ==, !=, <, <=, >, >=
    - Strings: ==, !=, StartsWith, Contains, EndsWith
- Examples: Register for notification so I can…
  - Launch my app when fast connectivity is available
  - Synchronize offline store when device is cradled
  - Perform behavior upon low battery notification
  - … or create your own!

Managed POOM

- Unified programming model for accessing Contacts, Calendar, Tasks
- Managed Interface
  - Collections of Contacts, Appointments, and Tasks
  - Restrictions and sorting of Collections
  - Create, edit, delete, copy, display, compare, change notification on individual Items
  - Much easier to use than native!
  - Performance improvements
  - Managed POOM and MAPI share the namespace
- Native: Improved performance
- Usage scenario:
  - update contacts/calendar/tasks based on info from a web service or a system event
  - Integrate contacts/calendar/tasks into line of business apps

Managed POOM Sample Code

Adding a New Contact

```
OUtlookSession myOutlook = new
   OutlookSession();
Contact c = new Contact();
c.FirstName = “Shaq";
c.LastName = “Oneil";
c.Email1Address = “shaq@lakers.com”;
myOutlook.Contacts.Items.Add(c);
```

Contact Selector

demo

Demo

- Run app
- Displays contact list
- Add, edit
- Working with the on device contacts
- Sync device with AS: contacts transfer
- Try calling
- Try SMS

Managed MAPI

- Unified access to messaging
- Leverage Inbox functionality
- Managed Interface
  - Send Email message
  - Send SMS message
  - Intercept SMS message
  - Maybe: Access to folders and accounts
  - Performance improvements
  - Managed POOM and MAPI share the same namespace
- Native Interface: Performance improvements
- User Scenario:
  - Automated exchange of information, conversation/turn based programs
  - Using SMS as a programmatic message queue

Managed MAPI Sample Code

Sending an Email Message

```
EMailMessage m = new EMailMessage();
m.To.Add(new Recipient(“Shaq ONeil",
                       “shaq@lakers.com");
m.Subject =“Nice work last night!";
m.BodyText = “Pistons or Pacers?";
m.Send("ActiveSync");
```

Sending SMS Message: Native

```
HRESULT SendSmsMessage (HWND hWNd, SMS_ADDRESS smsDest, LPTSTR pMsg) {
    HRESULT hr;
    SMS_HANDLE smshHandle;
    TEXT_PROVIDER_SPECIFIC_DATA tpsd;
    SMS_MESSAGE_ID smsmidMessageID = 0;

    // try to open an SMS Handle
    hr = SmsOpen(SMS_MSGTYPE_TEXT, SMS_MODE_SEND, &smshHandle, NULL);
    if (hr != ERROR_SUCCESS)
        return hr;

    // Set up provider specific data
    tpsd.dwMessageOptions = PS_MESSAGE_OPTION_NONE;
    tpsd.psMessageClass = PS_MESSAGE_CLASS0;
    tpsd.psReplaceOption = PSRO_NONE;
    tpsd.dwHeaderDataSize = 0;

    // Send the message, indicating success or failure
    hr = SmsSendMessage (smshHandle, NULL, &smsDest, NULL, (PBYTE)pMsg,
                         lstrlen(pMsg) * sizeof (TCHAR),
                         (PBYTE) &tpsd, 12, SMSDE_OPTIMAL,
                         SMS_OPTION_DELIVERY_NONE, &smsmidMessageID);
    SmsClose (smshHandle);
    return hr;
}
```

Sending an SMS Message: Managed

```
SmsMessage sms = new SmsMessage();
sms.To.Add(new Recipient(sendTo.Text));
sms.Body = “Some Text”;
sms.Send();
```

Intercepting SMS: Native

```
//=============================================================================
// MonitorThread - Monitors event for timer notification
//
DWORD WINAPI MonitorThread (PVOID pArg) {
    TEXT_PROVIDER_SPECIFIC_DATA tpsd;
    SMS_HANDLE smshHandle = (SMS_HANDLE)pArg;
    PMYMSG_STRUCT pNextMsg;
    BYTE bBuffer[MAXMESSAGELEN];
    PBYTE pIn;
    SYSTEMTIME st;
    HANDLE hWait[2];
    HRESULT hr;
    int rc;
    DWORD dwInSize, dwSize, dwRead = 0;

    hWait[0] = g_hReadEvent;        // Need two events since it isn't
    hWait[1] = g_hQuitEvent;    // allowed for us to signal SMS event.

    while (g_fContinue) {
        rc = WaitForMultipleObjects (2, hWait, FALSE, INFINITE);
        if (!g_fContinue || (rc != WAIT_OBJECT_0))
            break;
        // Point to the next free entry in the array
        pNextMsg = &g_pMsgDB->pMsgs[g_pMsgDB->nMsgCnt];

        // Get the message size
        hr = SmsGetMessageSize (smshHandle, &dwSize);
        if (hr != ERROR_SUCCESS) continue;

        // Check for message larger than std buffer
        if (dwSize > sizeof (pNextMsg->wcMessage)) {
            if (dwSize > MAXMESSAGELEN)
                continue;
            pIn = bBuffer;
            dwInSize = MAXMESSAGELEN;
        } else {
            pIn = (PBYTE)pNextMsg->wcMessage;
            dwInSize = sizeof (pNextMsg->wcMessage);
        }
        // Set up provider specific data
        tpsd.dwMessageOptions = PS_MESSAGE_OPTION_NONE;
        tpsd.psMessageClass = PS_MESSAGE_CLASS0;
        tpsd.psReplaceOption = PSRO_NONE;
        tpsd.dwHeaderDataSize = 0;
        // Read the message
        hr = SmsReadMessage (smshHandle, NULL, &pNextMsg->smsAddr, &st,
                             (PBYTE)pIn, dwInSize, (PBYTE)&tpsd,
                             sizeof(TEXT_PROVIDER_SPECIFIC_DATA),
                             &dwRead);
        if (hr == ERROR_SUCCESS) {
            // Convert GMT message time to local time
            FILETIME ft, ftLocal;
            SystemTimeToFileTime (&st, &ft);
            FileTimeToLocalFileTime (&ft, &ftLocal);
            FileTimeToSystemTime (&ftLocal, &pNextMsg-
          >stMsg);

            // If using alt buffer, copy to std buff
            if ((DWORD)pIn == (DWORD)pNextMsg->wcMessage) {
                pNextMsg->nSize = (int) dwRead;
            } else {
                memset (pNextMsg->wcMessage, 0,
                        sizeof(pNextMsg->wcMessage));
                memcpy (pNextMsg->wcMessage, pIn,
                        sizeof(pNextMsg->wcMessage)-2);
                pNextMsg->nSize = sizeof(pNextMsg-
          >wcMessage);
            }
            // Increment message count
            if (g_pMsgDB->nMsgCnt < MAX_MSGS-1) {
                if (g_hMain)
                    PostMessage (g_hMain, MYMSG_TELLNOTIFY,
          1,
                                 g_pMsgDB->nMsgCnt);
                g_pMsgDB->nMsgCnt++;
            }
        } else {
            ErrorBox (g_hMain, TEXT("Error %x (%d) reading
          msg"),
                      hr, GetLastError());
            break;
        }
    }
    SmsClose (smshHandle);
    return 0;
}
```

Intercepting SMS: Managed

```
Message rule = new
   MessageRule(“Company.Rule”, PostProcess.Delete);
rule.MessageFilter = new
   MessageFilter(MessageField.Body, “Token:”,
                 ComparisonType.StartsWith, true);
rule.Install(@“\Program Files\MyApp\App.exe”, “”);
rule.MessageReceived += new
   MessageRuleEventHandler(rule_MessageReceived);
```

Managed Telephony

- Managed access to basic phone functionality
  - Interface does not map directly to TAPI/exTAPI
  - Make phone calls (brings up dialer UI)
  - Maybe: Access to call log
    - Call Log Collection: missed, incoming, outgoing calls
  - Status/Notification
    - Voice call information (call state, duration, etc)
    - Phone/radio state (augmented by Notifications Broker)

Managed Telephony
Make a Phone Call

```
Phone myPhone = new Phone();
myPhone.Talk("+14259994444");
```

SMS & Telephony

demo

Demo

(if fixed…)

- Run app
- Displays contact list
- Try calling
- Try SMS

Others

- Direct3D And DirectDraw Mobile
  - API based on the desktop/Windows CE
  - Standardized access to 2D HW acceleration
  - Targeted: Video playback & 2D content
  - Footprint reduced: ~400K to ~200K
  - Architected to support rotated displays
- Picture & Imaging
  - Common dialog to browse and select an image from your device/media card

Picture Picker

demo

Demo

- Run picpick
- Displays My Pictures
- Navigates
- One line of code

Improving Abilities

"I am responsible of N thousand devices. Put me in charge."

Building tools for manageability, security, performance:

- **Configuration Manager & device management**
- **Unified Installer**
- **Pocket Watson**
- **Persistent storage**

Device Management

- Configuration Manager
  - Centralized way to add, update, query device configuration information via WAP-based XML
  - Configuration Service Providers
  - Native interface: no change since 2nd Ed.
  - Managed Interface
    - Wrapper around native API
- OMA DM compatible client on phone devices

Using Config Manager

```
XmlDocument configDoc = new XmlDocument()
configDoc.LoadXml(
    "<wap-provisioningdoc>"+
    "<characteristic type=\"BrowserFavorite\">"+

    "<characteristic type=\“Windows Mobile Team
    Blog\">"+
    "<parm name=\"URL\“
    value=\"http://blogs.msdn.com/windowsmobile\"/>"+
    "</characteristic>"+
    "</characteristic>"
    +"</wap-provisioningdoc>");

XmlDocument ResultDoc =
    ConfigurationManager.ProcessConfiguration(
                                configDoc, false);
```

Configuration Manager

demo

Demo

- Run IE – No blog link
- Run dmtest
- Run IE – Now has blog link
- What else can be configured?

Unified Installer

- WM 2003: PocketPC and Smartphone have different Installer versions
- Future version: Unified Application Installer
  - Crack CAB, check signature, extract files
  - Backwards Compatible
    - Install CABs built for PPC 2002/2003, SP 2002/SP 2003
  - Supports additional CAB types
    - WAP, combination WAP/.DAT
  - Supports pIE installs, silent (signed), prompt mode (signed/unsigned), external storage device
  - Integrate with Configuration Manager

Pocket Watson

Return crash data and logs to Microsoft and partners

- User permission
- Leverages existing Desktop Watson servers and reporting infrastructure
- How it works
  - Unhandled exceptions trapped
  - CAB file created on the device
  - CAB file sent to Desktop Watson servers
  - Web-based access to data

Persistent Storage

- Flash File System
  - On PocketPC there was none (RAM based file system)
  - On Smartphone 2002, it was “\IPSM”
  - On Smartphone 2003, it is “\Storage”
  - On Windows Mobile Future…
  - Use SHGetSpecialFolderPath!

```
#ifdef CSIDL_APPDATA
  SHGetSpecialFolderPath(NULL, szFolderPath,
                         CSIDL_APPDATA, TRUE);
#else
  // Not defined for previous/current generation PPC
```

Windows Mobile Resources

Get Tools & Resources:
Windows Mobile Developer Portal

Technical Support:
- **Tools and SDKs with emulators**
- **Technical articles and whitepapers**
- **Developer community**

Marketing Support:
- **Monthly newsletters**
- **Case studies**

More Support:
Windows Mobile Solution Partner Program

Technical Support:
- **Exclusive expert columns**
- **Early access to SDKs**
- **Access to beta programs**

Marketing Support:
- **PR support**
- **Ongoing promotions for devices**

Go To Market:
Mobile2Market and Certification

Technical Support:
- **“Designed for Windows Mobile” certification testing**
- **Free technical support incident**

Marketing Support:
- **“Designed for Windows Mobile” logo on packaging & promotions**
- **Increased promotion to retailers and distribution partners**

Go to:
http://www.microsoft.com/windowsmobile/developer
http://blogs.msdn.com/windowsmobile

Community Resources

Attend a free chat or web cast
http://www.microsoft.com/communities/chats/default.mspx
http://www.microsoft.com/usa/webcasts/default.asp

List of newsgroups
http://communities2.microsoft.com/communities/newsgroups/en-us/default.aspx

MS Community Sites
http://www.microsoft.com/communities/default.mspx

Locate Local User Groups
http://www.microsoft.com/communities/usergroups/default.mspx

Community sites
http://www.microsoft.com/communities/related/default.mspx

Session Evaluation

Please fill out a session evaluation on CommNet

Thank You!

Microsoft Tech·Ed 2004

Microsoft®

Your potential. Our passion.™

Microsoft Tech·Ed 2004